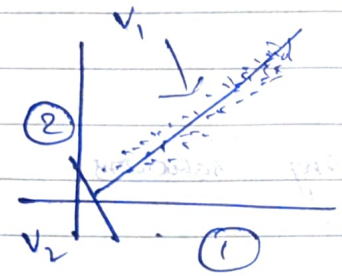
⇒ used for dimensionality reduction — unsupervised algorithm

Basics of Principal Component analysis :-



⇒ New are used.

⇒ PCA helps to reduce overfitting

$PC_1 \geq PC_2 \geq PC_3 \geq PC_4$

PC_1 have higher priority $> PC_2 > P$

Orthogonal property :-

⇒ PC_1, PC_2, PC_3 independent of each other

PCA numericals :-

$X = (2.5, 0.5, 2.2, 1.9, \ldots)$

$Y = (2.4, 0.7, 2.9, 2.2, 3.0 \ldots)$

1) find $\bar{x}$, $\bar{y}$

2) → Covarience matrix

2) $C = \begin{bmatrix} Cov(x,x) & Cov(x,y) \\ Cov(y,x) & Cov(y,y) \end{bmatrix}$

eg :- $Cov(x,y) \Rightarrow \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1}$

$\Rightarrow \frac{(x_i - \bar{x})(y - \bar{y})}{(n-1)}$

$\Rightarrow \boxed{C - \lambda I = 0}$, $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

3) Q.E → two roots λ_1, λ_2

v = eigen vector.

$Cv = \lambda v$

$x_1 = ?$

$x_2 = ?$

eg:- $X_2 = 0.92194\, Y_2$

$\Rightarrow \begin{bmatrix} 0.92194 \\ 1 \end{bmatrix} \Rightarrow \sqrt{(0.92194)^2 + (1)^2} = 1.3601$

$\Rightarrow 1.3601 \Rightarrow \left[\frac{0.92194}{1.3601}, \frac{1}{1.3601} \right]$

Regression analysis :- Supervised learning.

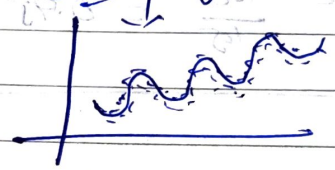
→ Dependent & independent variables

→ finds relations b/w dependent, independent.

→ ⇒ finds outliers (extremity)

→ ⇒ Multicollinearity ⇒ (independent variables collinearity)

→ underfitting, over-fitting



Training

Supervised ↓ faces Problem during training set

↓ Problem arises during test set. Supervised

Linear Regression

Dependent variables are continuous in nature.

→ linear R/ship (Dept, indep)

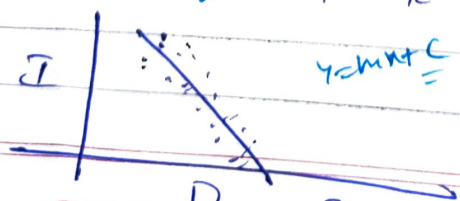
I-I, I-D

Simple linear R

M-I, M-D

Muti linear R

$y = \beta_1 + \beta_2 X_2 + \dots \beta_k X_k + \varepsilon$



$y = mx + c$

ε = error

β = coefficient of regression

Logistic Regression.

⇒ (True, success) (False, failure)

outcome = DV

Best fitting Model for I & D variable Relationships.

⇒ I.V can be continuous or Binary.

Used in ML.

Deals with probability.

Measure R/w dependent, independent variable.

Confusion Matrix (Predicted, Actual Values)

Tot. Samples

	Doctor	Predicted	
	165	NO	Yes
Actual	NO	TN	FP
	Yes	FN	TP

T-I 60 / 105

T-II 55 / 110

$$\text{Accuracy} = \frac{TP+TN}{\text{Total}}$$

$$\Rightarrow \frac{(100+50)}{165}$$

$$= 0.91$$

Error = 1 - Accur.

$= 0.09$

$$\text{Precision} = \frac{TP}{\text{Predicted yes}}$$

$$= \frac{100}{105} = 0.95$$

$$\text{Recall} = \frac{TP}{\text{actual yes}}$$

$$\Rightarrow \frac{100}{110} = 0.64$$

Dimensions ⇒ Features, attributes.

M_1	M_2	M_3	M_4	M_5	M_6	M_7
2	4	8	10	12	100	1000

accuracy

Threshold

Irrevelent feature

Eg:- Sphere Eatable Play Red

1) ✓ ✗ ✗ ✗

yellow

2 ✓ ✓ ✓ ✗

Threshold

Dimensity Selection is important.

* Managing missing Features :- (1) Remove the Record
Predict Missing Value. ← (2) Create Sub-model
(3) Automatic Strategy
↳ mean
↳ median
↳ Mode.

Create Sub-model :-
↳ Predict the Value not actual.

* Managing Categorial data :-
eg:- Weather (Sunny, Cloudy, Rain) (M=3)

1 One hot encoding (M=3)

Weather	S	C	R
S	1	0	0
S	1	0	0
C	0	1	0
R	0	0	1

Dummy Coding Scheme (M-1) (M=2)

Weather	C	R
S	0	0
S	0	0
C	1	0
R	0	1

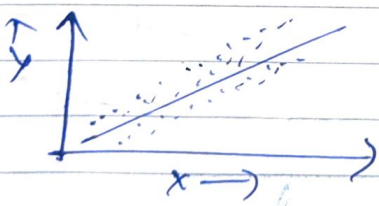
Effect Coding Scheme.

Weather	C	R
S	-1	-1
S	-1	-1
C	1	0
R	0	1

Vector representation. (M-1)
Binary representation

---

linear Regression :- $y = mx + c$
dependent variable are continuous in nature.
Regression analysis is used to find relationship between independent & dependent variables.



$y = 0.9 + 1.2x_1 + 2x_2 + 4x_3 + 1x_4$

Simple linear regression

$y = \alpha_0 + \alpha x_1$

↓ dependent y ↑ independent

Co-efficient

$y = mx + c$

↓ Intersect point.

Multiple linear equation

$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha \ldots \alpha_m x_m$

α_i = Reg. Coeff → Imp

x_i = Independent Vari,

y = Dependent Vari

α = Importance of the attribute independent

α :- Importance of the attribute $\left(y = \frac{1}{1+e^{-x}}\right)$
y highly depends on α

---

logistic Regression :- $e = 2.718$

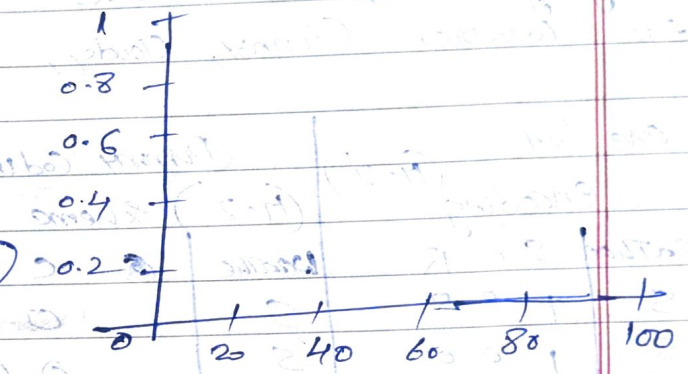
$$y = \frac{1}{1+e^{-x}}$$

(Sigmoid fn)

→ fraud detection
→ disease diagnosis
→ Emergency detec
→ Spam, No spam

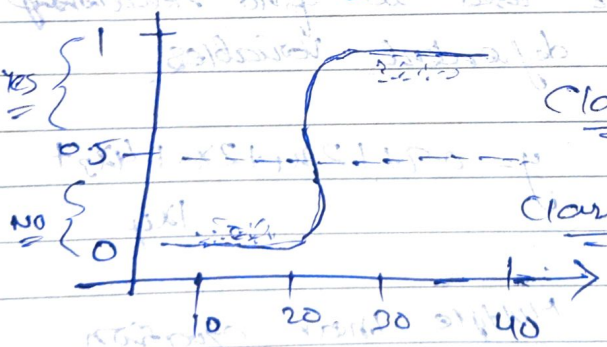
Time	Clicked on AD
68.95	NO
80.23	NO
69.45	NO
74.15	NO
50.0	yes
55.5	yes
80.0	NO
70.5	NO

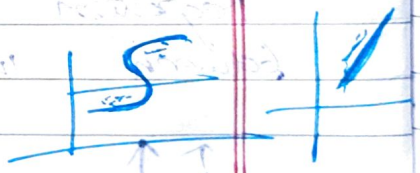
Categorical data



In classification logistic Regression is highly used

Sigmoid helping to change independent variable into useful Probabilistic expression ranging 0-1 w.r.t dependendent variable.





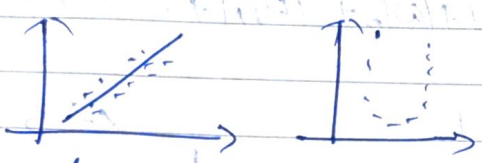
$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$

Polynomial regression:

$y = \alpha_0 + \alpha_1 x_1$

Multiple linear $y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \ldots$

$y = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$



linear (1°) — No linear (2°)

0 degree $= y = C$

1 degree $= y = mx + C$

2 degree $y = ax^2 + bx + C$

$$\boxed{y = \alpha_0 + \sum_{i=1}^{m} \alpha_i x_i + F_p}$$

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x^2 + \alpha_3 x^3 + \ldots + \alpha_n x^n$$

F_p = polynomial functions.

polynomial regression $\Rightarrow$ $\boxed{Y = \alpha_0 + \sum_{i=1}^{m} \alpha_i x_i + F_p}$

Ridge regressions $\rightarrow$ to reduce overfitting & underfitting

Regularization $\rightarrow$ Ridge
$\rightarrow$ lasso

Ridge $\boxed{R = \text{Loss} + \alpha \|w\|^2}$ (penalty)

$y = 0.4 + 1.2x_1 + 20x_3 + 39x_3$

$y = 0.9 + 0.7x_1 + .2x_2 + 5x_3$

Ridge regression Reduces Cost of Computation.

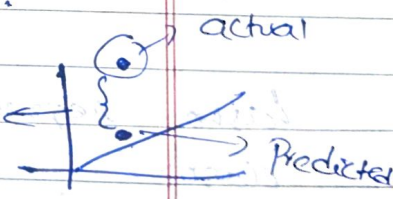
$\rightarrow$ Reduces Complexity of Model

$\rightarrow$ Makes Computation easier.

$\alpha \|w\|^2$

$R = \text{Loss} + \text{penalty}$.

$\alpha \uparrow$ cost $\downarrow$

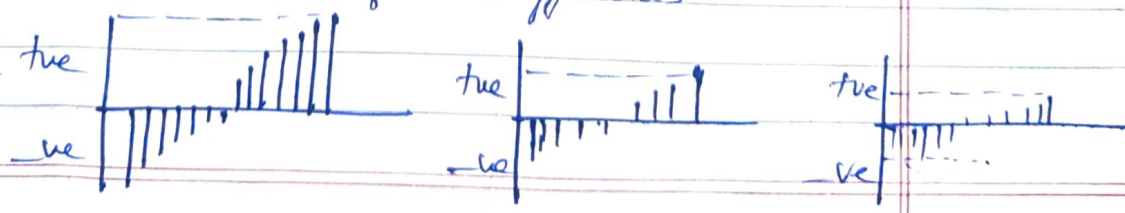


α = Constant

$\|w\|^2 = w_1^2 + w_2^2 + \ldots w_n^2$

w = vector of coefficients

α = coefficient



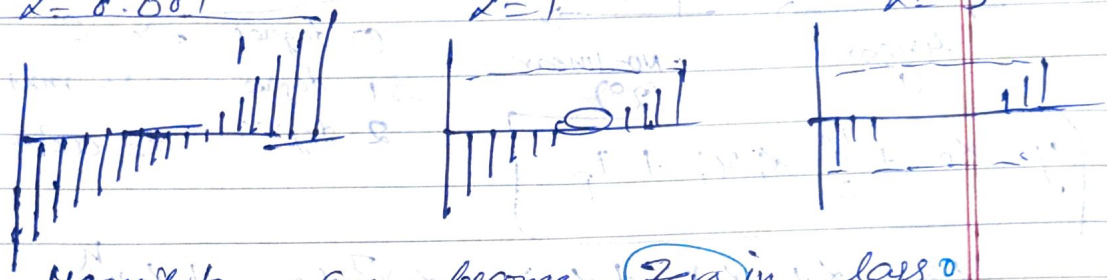
$\alpha = 0.01$ $\alpha = 1$ $\alpha = 2$

$\alpha \, ||w||$ loss cost:

Lasso:- Loss + (PENALTY)

$||w|| = |w_1| + |w_2| + |w_3| + |w_4| + \dots |w_n|$

$\alpha = 0.001$ $\alpha = 1$ $\alpha = 5$



Magnitude can become zero in lasso whereas it can't in ridge.

lasso is optimized. → attributes can be neglected

Computation is more optimize → decreases attribute

→ features can be selected.

---

* ELASTIC NET Regression:-

Ridge R = Loss + $\alpha \, ||w||^2$

Lasso R = Loss + $\alpha \, ||w||$

Elastic Net R = Loss + $\alpha_1 \, ||w||^2 + \alpha_2 \, ||w||$

Ridge regression:- Multi collinearity Problem

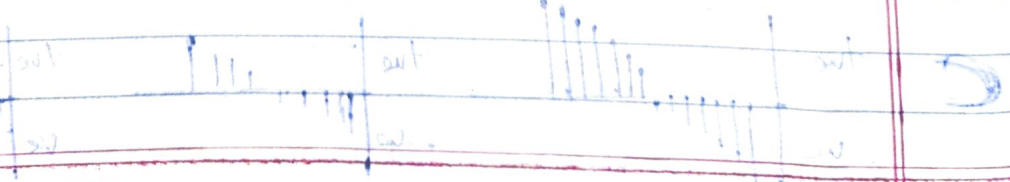
↳ attribute dependency changing

Lasso regression:- feature selection.

elastic = (Ridge + loss)

=)

Elastic Net = Hybrid of Ridge & Lass



Date
Page

* Conditional probability :-

$$\boxed{P(A/B) = \frac{P(A \cap B)}{P(B)}}$$

Bayes Theorem :-

$$P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$$

likelihood

prior

Marginal

Posterior

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \qquad P(B/A) = \frac{P(B \cap A)}{P(B)}$$

$$P(A \cap B) \Rightarrow P(A/B) \cdot P(B) \Rightarrow P(B/A) \cdot P(A)$$

$$\Rightarrow P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

Naive Bayes classifier :- Supervised, for classification Tasks.

=>

Fruit	yellow	Sweet	long	Total
Orange	350	400	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

$$\Rightarrow P(\text{yellow} / \text{orange}) \Rightarrow \frac{P(\text{orange}/\text{yellow}) \cdot P(\text{yellow})}{P(\text{orange})}$$

$$\Rightarrow \frac{\left(\frac{350}{800}\right)\left(\frac{800}{1200}\right)}{\frac{650}{1200}} = 0.53$$

$P(S/O) = 0.69, \quad P(L/O) = 0$

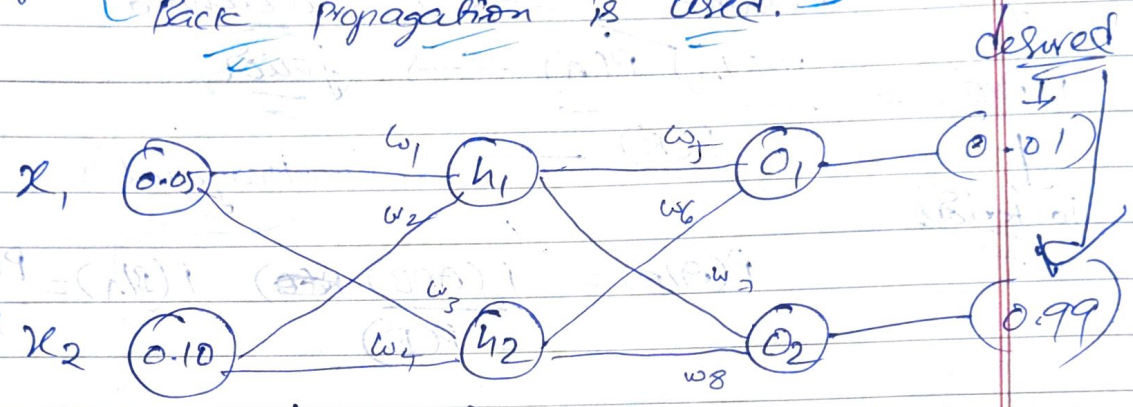
$P(\text{Fruit} / \text{orange}) = 0.53 \times 0.69 \times 0 = 0$

Fruit = Banana

B - bias factor

Back propagation / Backward propagation of error. (Neural Network used)

(If [actual output = desired output] Back propagation is used.)



$b_1 = (0.35)$ $b_2 = (0.60)$

$h_1(in) = w_1x_1 + w_2x_2 + b$, $h_1(out) = \frac{1}{1+e^{-h_1(in)}}$

$h_2(in) = ?$ $h_2(out) = ?$

Sigmoid function $= \frac{1}{1+e^{-h_1(in)}}$

Sigmoid function is used.

$o_1(in) = ?$ $o_1(out) = ?$

$o_2(in) = ?$ $o_2(out) = ?$

O_1, O_2 = actual output

$E_{Tot} \Rightarrow \Sigma\left(\frac{1}{2}(Target - O/P)\right)^2$

$EO_1 = 0.274$, $EO_2 = 0.023$

$E_{total} = EO_1 + EO_2$

at final weights are (adjusted) by

Trail & error method.

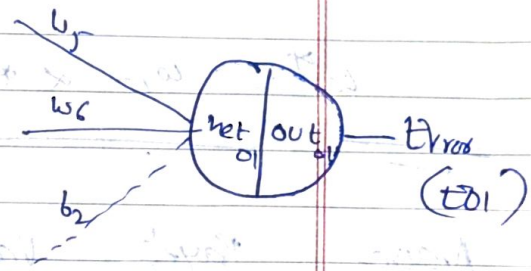
Backpropagate

$E_{Total} = 0.298371109.$

(hidden layer)

⇒ Input layer, (inner layer), output layer

$$\Rightarrow \frac{\partial E_{Total}}{\partial w_5} = \frac{\partial E_{Total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$



$$\Rightarrow \frac{\partial E_{Total}}{\partial out_{o1}} \Rightarrow Out_{o1} - Target_{o1} = 0.741365$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1}(1 - Out_{o1}) = 0.186815602$$

$$\frac{\partial net_{o1}}{\partial w_5} = Out_{h_1} = 0.59326992$$

$$\Rightarrow \frac{\partial E_T}{\partial w_5} \Rightarrow 0.082167$$

learning rate

$$\Rightarrow w_5^* = w_5 - \alpha \times \left(\frac{\partial E_{Total}}{\partial w_5}\right) = 0.35069997 6$$

adjusted cost of w_5

(partial derivative)

⇒ Same method the hidden layer.

$$\Rightarrow \frac{\partial h_{out}}{\partial w_1} \Rightarrow \frac{\partial h_{out}}{\partial E_{Total}} \times \frac{\partial out_{h_1}}{\partial net_{h_1}} \times \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{Total}}{\partial h_{out}} \Rightarrow \frac{\partial E_{o1}}{\partial out_{h_1}} + \frac{\partial E_{o2}}{\partial out_{h_2}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h_1}} + \frac{\partial E_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o2}}{\partial out_{h_1}}$$

Combination:

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}(1 - out_{h_1})$$

$$net_{h_1} = w_1 x_1 + w_2 x_2 + b_1 x_1 \qquad \frac{\partial net_{h_1}}{\partial w_1} = x_1 = 0.05$$

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}(1 - out_{h_1})$$

$$w_1^* = w_1 - \alpha * \frac{\partial E_{total}}{\partial w_1}$$

Naive Bayes Varient.

Discrete Cont for Continuous

Bernouli

Bernouli distribution, Binary Values (0/1)

$x = 1$ (success)

$x = 0$ (Failure)

x has Bernouli distribution.

$$P(X = x) = p^x (1-p)^{1-x}$$

$$P(x) = \begin{cases} p & \text{if } x = 1 \\ q & \text{if } x = 0 \end{cases}$$

Multinomial

Word in Text Doc. → frequency Test.

* multinomial distribution.

BG	O	A	B	AB
P	0.44	0.42	0.1	0.04

$$P(X_1 = x_1 \ldots X_k = x_k)$$

$$\boxed{\frac{n!}{x_1! \ldots x_k!} p_1^{x_1} \ldots p_k^{x_k}}$$

$$\frac{6!}{1!2!2!} 0.44^1 \, 0.42^2 \, 0.1000^2$$

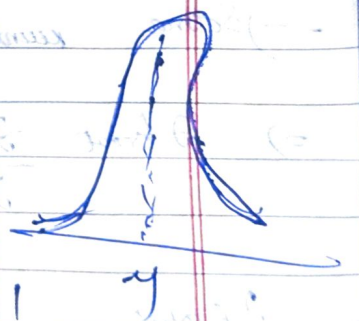
Guassian

→ guassian distribution

PDF :- Probability density function

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

$$-\infty < x < \infty$$



μ = mean

σ^2 = Varience

σ = ~~Varien~~ S.D

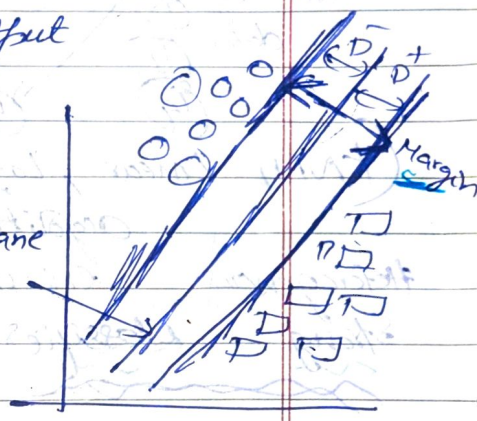
Normal distribution is used.

SVM :- Support Vector Machine (SVM)

SVM :- Supervised learning

Quadrilateral □
Circles ○ ⇒ Model Training ⇒ Prediction ← New data
↓
output

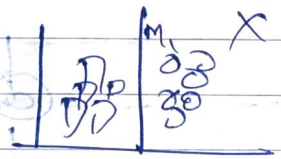
SVM used for
Classification & Regression.

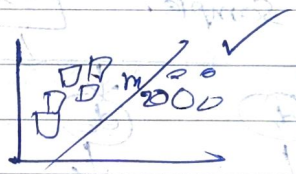


○, □ are classes in SVM

On the plane near to the D⁻, D⁺ line (touching)

(Maximum width should be high) → if new points adds



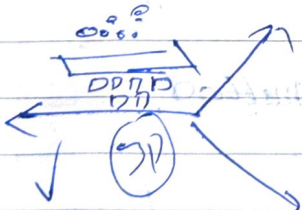


$m_2 > m_1$

∴ decreases error rate.

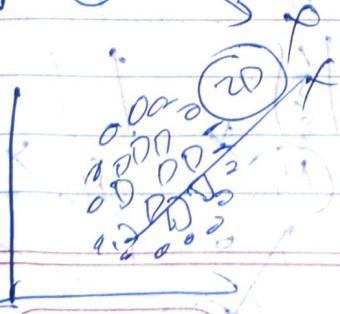
Non-linear SVM & kernel function.

Kernalization



LD → kernal ⇒ HD
↓
Non Seperable
↓
can't be classified.

↓
Seperable
↓
Can be classified

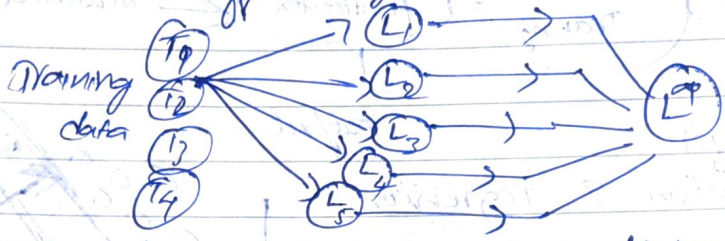


Ensemble learning :-

→ Different algorithm

→ Different training data set.

ensemble diff. algorithms.



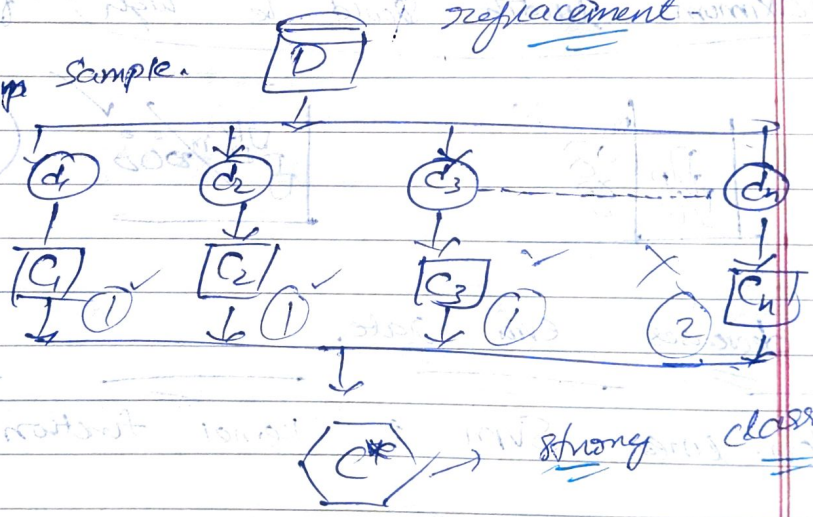
(KNN, Linear, Logistic, SVM, Naive Bayes, Decision Tree) algorithms

Heterogenous ensemble.

Strong classifier → (ensembling learning)

* Bagging method :- random samplings with replacement

Bootstrap sample.



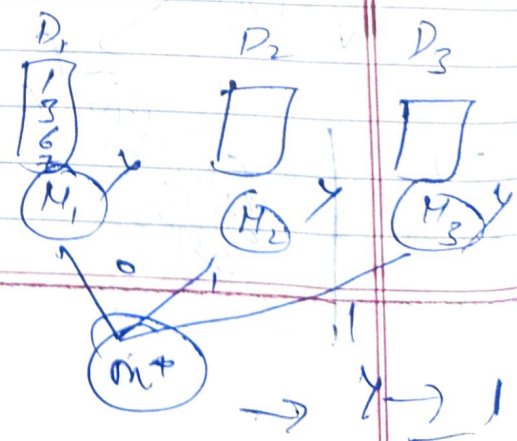
→ strong classifier.

different samples, different algorithms.

Boosting :- Sequential ensemble method.

n instances,

$1\ \omega_1$
$2\ \omega_2$
$3\ \omega_3$
$\vdots$
$n\ \omega_n$



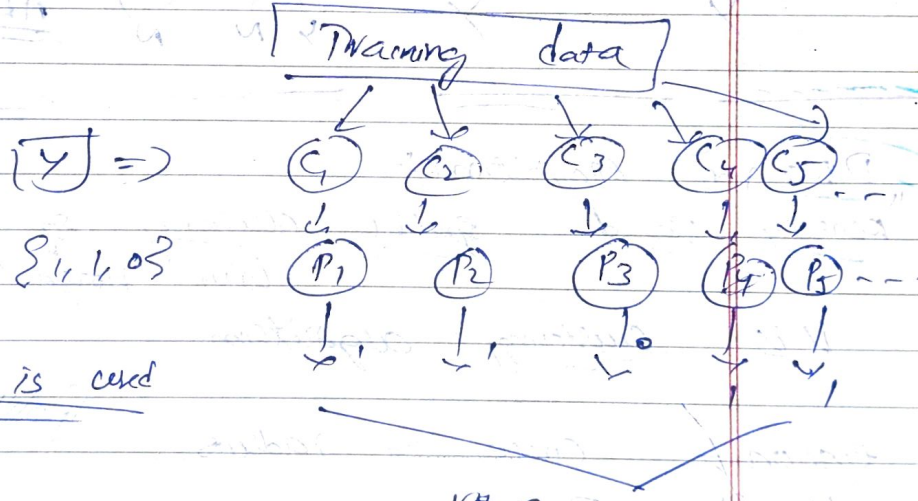
M_1 operations:- 1) M_1 Construction with equal weights
2) prediction of Total data set
3) comparing with classes
4) Misclassified elements weights are increased.

use altered weights in D_2 and construct model.
follows same procedure like M_1

4) weights are increased in Boosting.

Voting classifier :- Hard voting & Soft voting

Hard voting :- Majority



$Y = \text{mod}\{1, 1, 0\}$
$\Rightarrow 1$
Mod is used

Final prediction. P4 '1'

Soft voting :- $C_1 \Rightarrow 0.9, 0.1$
$C_2 = 0.8, 0.2$
$C_3 = 0.4, 0.6$

probabilities of each prediction in each model. $\Rightarrow \frac{0.9 + 0.8 + 0.4}{3} = 0.7$
pick, The highest total probability. $= 0.3$

Random Forest :-

Kind of ensembie classifier. used Decision tree algorithm.

More Decision Trees are made, Choosing randomly.

OD → BD

original data — Bootstrap data

Replicates allowed

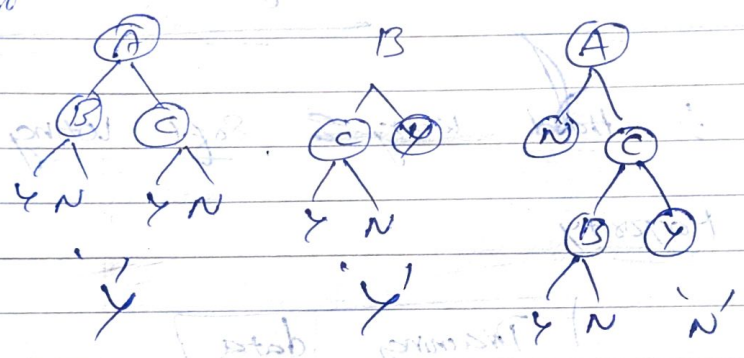
OD

	A	B	C	T
1				Y
2				N
3				Y
4				N

BD

	A	B	C	T
1				Y
4				N
4				Y
2				N

Diff erent trees



ABC $\boxed{Y}$

DBSCAN Algorithm :-

"Density Based Spatial clustering of Application with noise"

It is clustering algorithm.

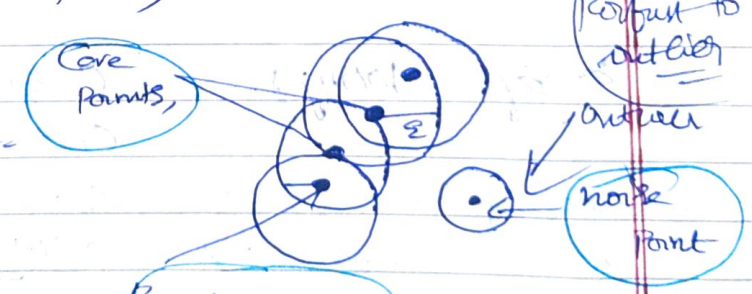
Robust to outliers

Epsilon :- Circle ka radius

Min Pts = 3.

Core point Center point,

Directly density reacheable.



two datapoints are near then they are said to be on same cluster, + neighbour.

K- means algorithm Clustering algorithm.

euclidean distance = $\sqrt{(x_0-x_c)^2+(y_0-y_c)^2}$

take 2 two points k_1, k_2 weath, Height, weight.

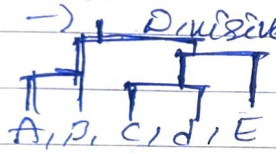
Calculate distance for 3rd point near then Kluster update the centeroid

Hierarchical clustering :-

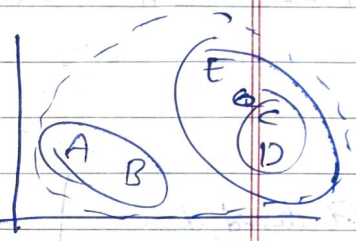
(Dendograms)

-> Agglomerative

-> Divisive.

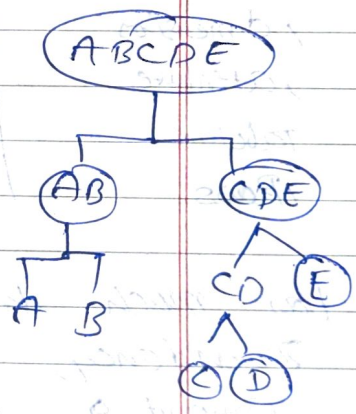


=> Hierarchical Clustering.



Agglomarative => Bottom -> Up

Divisive => up- Bottom



=> Agglomarative clustering :

indclared :-

*, find min dist, from matrix,

, next from cluster,

, calculate distance, take min (d, [d_3, d_4])

min ((d_1, d_3), (d_1, d_4))

Agglomarative :- -> Single linkage -> Min

-> Complete linkage. - max

Recommendation System.

- Content based. — Past watched movies by u_1
- Collaborative filtering.

u_1 — u_2

5 M_1 — 5 M_1

4 M_2 — 5 M_2

4 M_3 → recomended by user similarity,

Data in machine learning.

Data

Structured data	Unstructured data	Big data
, Column	Images, videos	flipcart recommendation from every data,
, dimension	Messages	
, attribute	Actions	People also bought
Tabular.		decoding.
, Rows		

How much data do I need?

Redundency, underdated data for ~~huge~~ huge amount of data.

X ÷ attribute => 10X = needed for training

(clustering, dimensity reduction algorithm

→ 10,000 samples

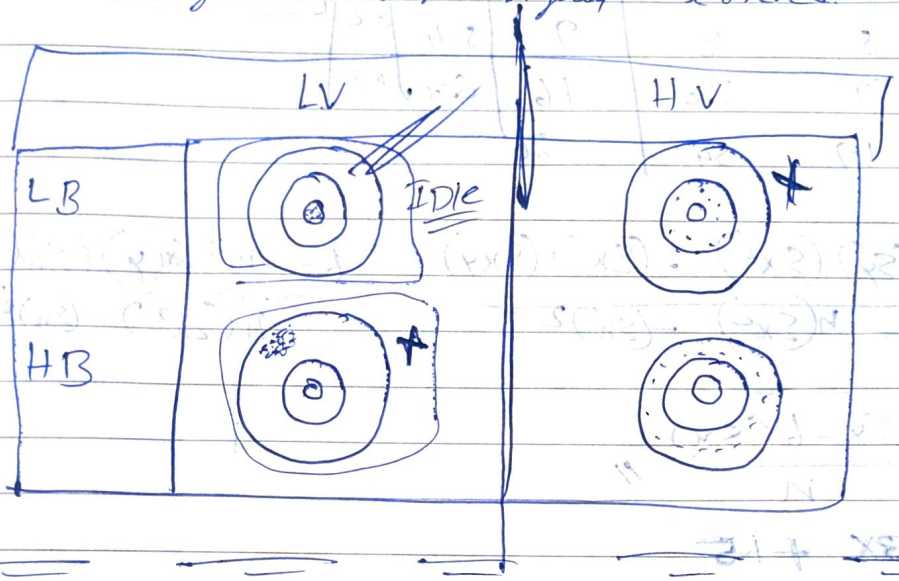
Regression, classification => 1,00,000 samples

Neural network, huge.

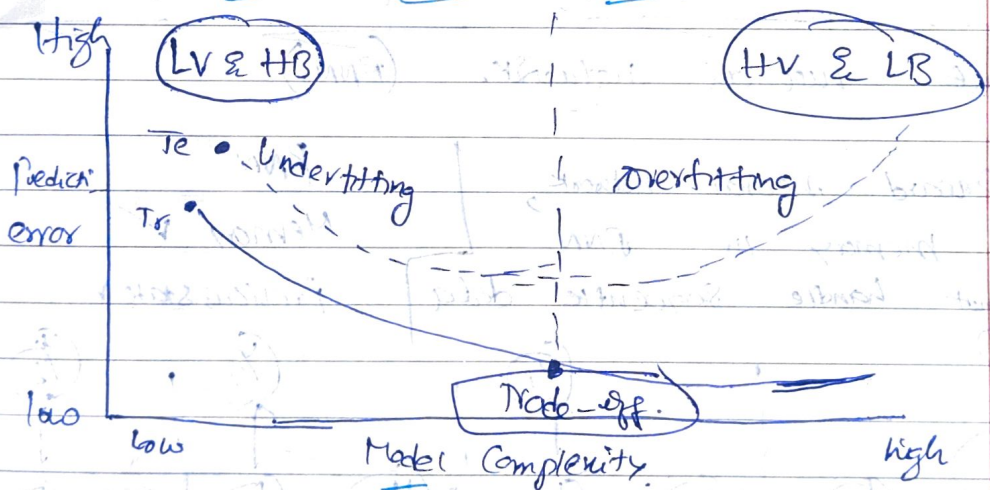
Predicted value
actual value

Bias & variance:-

* High Bias :- predicted value is far away
  low Bias :- predicted value is less near.
  Low variance :- low scattered
  high variance :- highly scattered



⇒ Bias - variance trade off



Linear regression : Solved Sum :-

⇒

X	Y	XY	X^2	P	error
1	3	3	1	2.8	0.2
2	4	8	4	4.1	0.1
3	5	15	9	5.4	0.4
4	7	28	16	6.7	0.3
10	19	54	30		

$\boxed{Y = bx + a}$

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2} \qquad b = \frac{n(\Sigma xy) + \Sigma x(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$a = \frac{\Sigma y - b(\Sigma x)}{n}$$

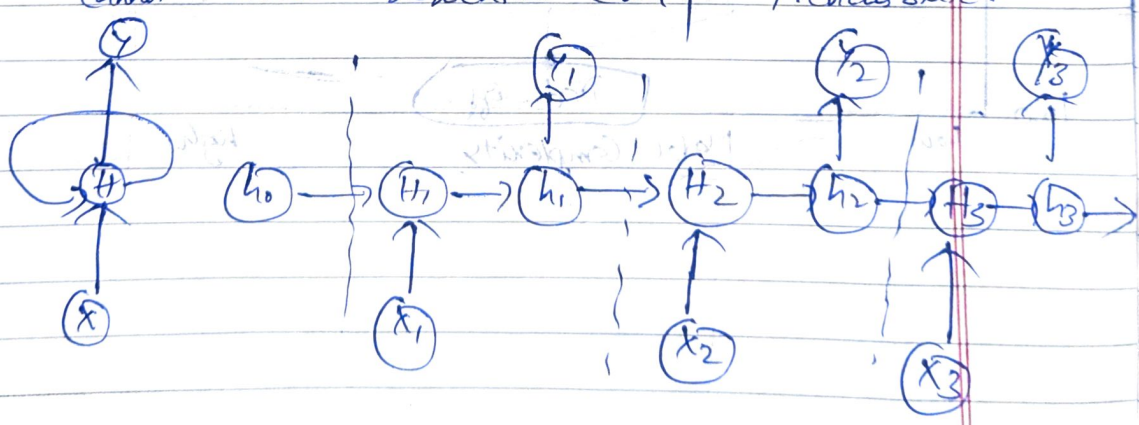
$Y = 1.3X + 1.5$

This is how we can find value of "Y"

Recurrent neural network. (RNN)

⇒ feed forward neural network,
NO memory in FNN
Cannot handle sequentie data

RNN
Memory ✓
previous state ✓



$h_t = = f(h_{(t-1)} X_t)$, $h_t = \tanh(W_{hh} h_{t-1} + W_{xh} X_t)$

$Y_t = W_{hy} h_t$

Can handle sequential data

$h_t = f(h_{t-1}, x_t)$

Same function is performed

$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$

activation function

W_{hh} = weight at previous hidden state

W_{xh} = weight at current input state.

$y_t = W_{hy} h_t$

=> W_{hy} => Weight at the output state

h_t => timestep at that h.

---

data.shape, from pandas import read_csv

data.dtypes => provides data types of each row

data.describe => mean, std, min,

Matplotlib used to visualization of data.

~~Plt.plot~~ (data) show, xlabel, ylabel, bar, plot.

from matplotlib import pyplot as plt

✓ Plt.title()

histograms ranges of values of the attribute.

✓ Plt.hist

✓ Plt.pie Plt.pie (a, labels = label, colors = cols,

✓ Plt.show()

startangle = 120,

moves the color column away ← explode = (0,0,0,0))

Plt.boxplot (attribute) => Boxplot => outliers, min, max, upper, lower quartile

Plt.show

Violin plot :- distribution of numeric data, ~~mean~~, median,

Comparison of various attributes

Normalization: Data preprocessing:

$$x' = \frac{x - min}{max - min}$$ normal value

rescale the features b/w (0-1)

Sklearn (preprocessing)

Standardization, $z = \frac{x - \mu}{\sigma}$

Important,

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(

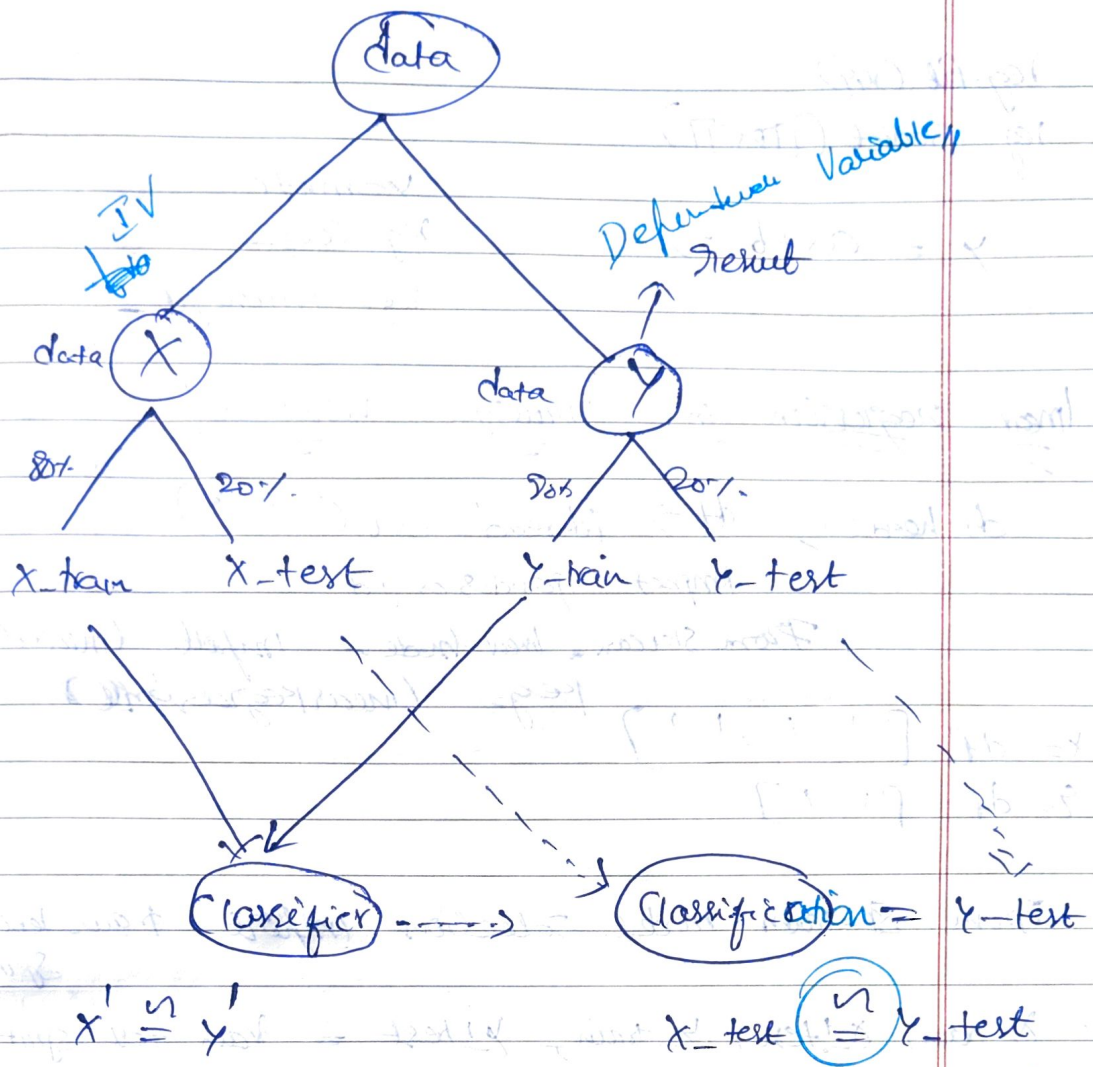
(X, Y, test_size = 0.3, rand_state = 10)

random_state = helps not to change data randomly during train, test runs

	A	B	C	D	E	F	R
train data			X_train				Y_train
test data			X_test				Y_test

X - Independent

Y - value dependent



linear Regression :- → Single variable
predict & classify → multiple variable.

⇒ Single variable :-

```
import matplotlib.pyplot as plt
plt.scatter (df ['distance'], df ['price'])
plt.scatter (df ['distance'], df['price'])

import matplotlib.pyplot as plt.

from sklearn.linear_model import LinearRegression.
~~clf = LinearRegression()~~
x = df [[ ]]
y = df [ ]
reg = LinearRegression()
reg.fit (x, y)
```

reg.fit (x,y)
reg. predict ([[25]])

$y = ax+b \Rightarrow$

y=mx+c
reg. coef –
reg. intercept –

Linear regression to multiple variables:

df.head(), df => pd.read_csv(" ")
import pandas as pd
From sklearn.linear_model import LinearRegression
Reg= LinearRegression()

x= df [' ' , ' ']
y= df [' ']

From sk-learn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split (x, y, test_size = 0.3, random_state = 10)

from sklearn.linear_model import LinearRegression

=> From sklearn.model_selection import train_test_split
From sklearn.linear_model import LinearRegression
x_train, x_test, y_train, y_test = train_test_split (x, y, test_size = 0.3, random_state = 10)

Reg = LinearRegression()

reg.fit (x_train, y_train)
reg.Predict (x_test)
reg.score (x_test, y_test)

Coeff = ?
Intercept = ?
y = ax + bx + c
=>

Intercept –
Coeff –

(0, 1), (yes, No) (T, F)

logistic regression: categorical value,

=> from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split
(df [['age']], df['sheet'], train_size=0.8,
random_state=10)

from sklearn.linear_model import LogisticRegression
reg = LogisticRegression()
reg.fit (x_train, y_train)
reg.predict (x_test)
reg.score (x_test, y_test)

---

from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
df = pd.read_csv (" ")

plt.scatter (df.rollno, df['marks'])
plt.xlabel ('rollno')
plt.ylabel ('marks')

km = KMeans (n_cluster=3)
predicted = km.fit_predict (df['rollno', 'marks'])

km = KMeans (n_cluster=3)
predicted = km.fit_predict (df ['rollno', 'marks'])

df['Cluster'] = Predicted
df1 = df [df.Cluster==0]
we can assign colors,
like scatter,

plt.scatter (df1.rollno, df1['marks'],
color='green')

Date / / Page

Scale = MinMax Scaler ()
Scale.fit (df['marks'])

df['marks'] = Scaler.transform (df['marks'])

Before performing Clustering perform preprocessing.

km.cluster_centers_

Time Series :- Level, Seasonality, Noise, Trend, Cyclic patterns,

Trend :- → up trend
→ down trend

Seasonality :- patterns,

Statistics :- Collect, organize, analyze & interpret & represent
→ for decision making.
→ Descriptive
- Measure of central tendency (Mean, Median, Mode)
- → MOD (Range, SD, variance) (Measure of Dispersion)
→ Inferential
- → Z-test
- → T-test
- → F-test
→ (Conclusion)

Source 1 →
Source 2 →
Source n →
[Raw Data] → Feature → Modelling → insights

↑ Clean & Transform

DSLC :- Data Science life cycle :-

→ BR :- Business Requirement

→ DA :- Data acquisition

(1) ⇒ Data processing ****

→ Data exploration :- understand the data visualization

→ Modelling

→ Deployement.

NDbrief

Sampling :- population :- Complete data set

↳ sample

Random, Systematic, Stratified samplings

Random sampling :- Randomly choosen

Systematic sampling :- every 10th record every 3rd, 4th line that

Stratified Sampling :-

Stratum :- Common Characteristics, like 10th no - even.

(1) Team	even Roll	→ to select
(2)	odd Roll	→ to selected

Point estimate & interval estimate

Point estimate :- (9) → mean, $\bar{x}$ mean = point → approximate point

"interval estimate" :- {100, 200} accuracy is high

Confidence interval CI = ?

$$\left(\bar{x} \pm z \frac{\sigma}{\sqrt{n}}\right)$$

Constant

80 → 1.282

85 → 1.440

90 → 1.645

95 → 1.960

99 → 2.576

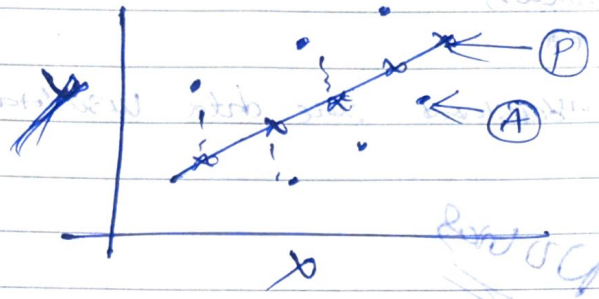
MOE :- Margin of error :- CI = 95

$$CI = \bar{x} \pm 2\frac{s}{\sqrt{n}}$$

$$\left(\frac{2s}{\sqrt{n}}\right) = MOE$$

MSE :- mean squared error

actual, estimate (predicted)



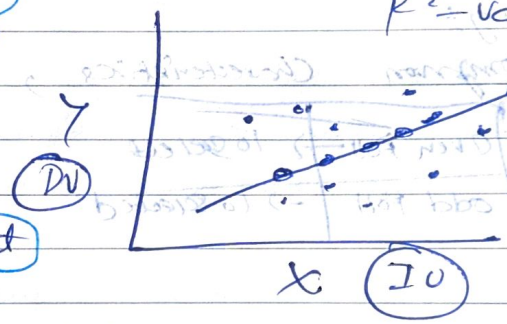
distance b/w them = error

$$\frac{\sum_{i=1}^{n} (P_A - \hat{P}_P)^2}{n}$$

$$\frac{\sum_{i=1}^{n} (P_A - P_P)^2}{n}$$

R-squared R^2

R^2 - value



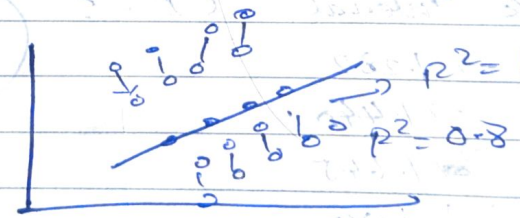
Reg. line best fit

$R^2 \rightarrow$ describes about best fit

$R^2 = (r)^2 \Rightarrow$ ~~coef~~ coefficient of differentiation

$$\frac{\sum_{i=1}^{n} (P_P - \bar{P})^2}{\sum_{i=1}^{n} (P_A - \bar{P})^2}$$

($R^2 \uparrow$ error $\downarrow$)

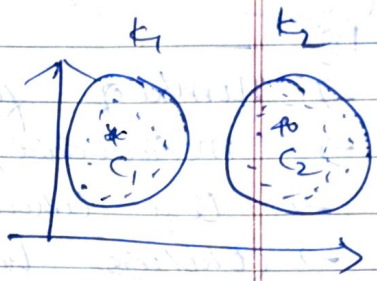


$R^2 = 1$

$\rightarrow (P = A)$

Random State : None , 0, 10

⇒ elbow method :

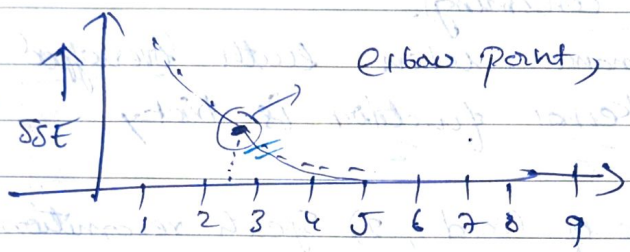


for $k=2$

$SSE_1 = \sum \text{Distance}(x-c_1)^2$

$SSE_2 = \sum \text{Distance}(x-c_2)^2$

$SE = SSE_1 + SSE_2$



elbow point play a major role in deciding the K → cluster

no. of

SVM (Support vector machine)

Pros

- → Number of features, Columns well more
- → medical data, image data, gene data
- → Best when classes are seperable
- → Outliers has less impact on svm,
- → used for binary Classification

Cons:-

Slow, time Consuming.
Poor performance when with overlapped classes
Selecting kernal function is tricky

Applications:- 1) Bags of word, 2) Speech recognition,
3) Classification of image,
4) medical analytics, 5) text classification

Naive Bayes

Pros	Cons
• Real time prediction, fast	• Each feature is independent on other.
• Scalable	→ Bad estimator
• Irrelevent features has less more than binary impact.	→ Training data should represent population well
• Multi-class prediction	
→ High dimensionality.	

Applications:- Sentimental analysis, Text classification
Spam filtering, recommendation system

Logistic Regression:

Pros:

- Simple, effective
- Scalling not required
- Tuning of hyper parameter not needed.

Cons:

- Poor performance on non linear data
- poor with highly correlated features
- not much powerfull
- High reliance on presentation of data

(*) Classification,

Multi-Class Classification binary is Preferred.

Student can pass & fail?

Customer taking loan predict?

Cancer detect yes/no: ?

(*) Random Forest:

- Random Forest decorelates trees,
- data size remains same but can have subset

  eg: [1,2,3,4,5,6] [1 2 3 3 4 5]
- overall error & varience gets reduced.
- good performance on imbalanced dataset
- handles huge amount of data
- good handling for missing data
- little impact of outliers
- No problem of overfitting
- Feature importance extraction

Cons:

Features should have predictive power.

tough to know what is happening

appears as black box

Application:

Credit card default, fraud customer, patients disease classify.

Recommendation for ecommerce.

Decision tree:

Pros

No need of normalization
Handles missing values
easy to explain
easy visualization
Automatic feature selection

Cons:

prone to overfitting
Sensitive to data
Higher time ↑ (training)

Application: identifies buyers for products, fault diagnosis in machines, frequent discount, frequent shopping

X G Boost: IMP

Pros:

→ Less feature engineering
→ Feature importance recognition
→ Fast to interpret
→ minimal impact of outliers
→ Handles large size dataset
→ good execution speed
→ good model performance
→ less prone to overfitting

Cons:

visualization is tough
Hard to tune
overfitting.

KNN: K - nearest neighbours

Pros

No assumption
Simple
Constant evolving
Multi class problem can be solved
one hyper parameter

Cons:

Slow for large datasets
Sensitive to outliers
can't deal missing values

Application: less feature, it dont know the shape, ~~classification~~ application

Classes can be seperated by line, ellipse, parabola

PANDAS:

- Flexibility of python
→ working with big data

import pandas as pd
df = pd.read_csv ("/ //")
→ df.head(3)
→ df.tail ( )
df = XLSX = pd.read_excel ('Pokemon_data.XLSX')

delimiter = '\t' used to show the .txt data
→ df.columns(c) ✗
→ df['Name'] [0:5]
→ df([['Name', 'Type 1', 'HP']])

→ df.iloc[1] (integer, location)

df.iloc[0:4] df.iloc[2, 1]
2nd row → 1st column

for index, row in df.iterrows():
print (index, row['Name'])

df => [df.iloc[:, 4:9].sum(axis=1)] = (Horizontal)
(axis=0) (Vertical)

(column, row)
[:, 4:9]

=> df.loc[df['Type 1'].str.contains('h', regex=True)]

Model Performance strictly -> XGBoost, neural network,

features are highly independent Naive Bayes
too many features, medium size SVM
linear relationship b/w IV, DV Then SVM, linear, logistic
No, assumption, dataset is small KNN

Missing Values:-

=> Missing values = ["na" "na", "--"]

data = pd.read_csv (' ', na_values = Missing values)